Software Engineering Fundamentals

- Lecture 5: Behavioural Models
  - Behavioral models
  - What are Interaction Diagrams?
  - Sequence Diagrams
    - concurrent processes
  - Collaboration Diagrams
  - Extra Notes

UML Models/Diagram types

1. Class
2. Object
3. Component
4. Composite structure
5. Use Case
6. Sequence
7. Communication
8. State
9. Activity
10. Deployment
11. Artifact
12. Package
13. Timing
14. Interaction overview

Structural

Behavioural

Behavioural models

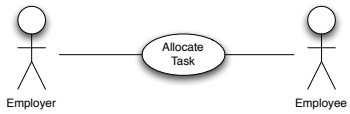
- In the next few lectures we will look at behavioural models.
- These describe how objects work
  - internally (state diagrams)
  - with each other (communication, sequence, activity)
- We will look at each of these four models

What are Interaction Diagrams?

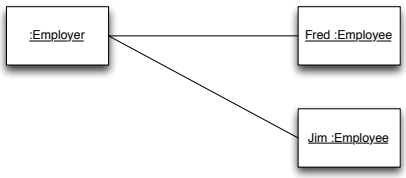
- Interaction Diagrams show how certain participants (or groups of objects) collaborate in some behaviour
- Generally an Interaction Diagram describes one Use Case
- Several types
  - Sequence Diagrams
  - Interaction Overviews
  - Communication Diagrams
  - Timing Diagrams
  - Interaction Tables

Communication diagrams

- Consider the problem of a company allocating a task to an employee...

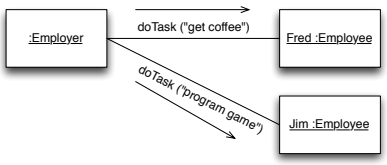


- An object diagram could be...

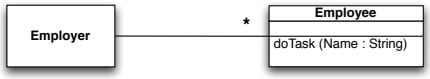


Communication diagrams

- We can transform this into a communication diagram by showing how the objects communicate to fulfill a use-case scenario.

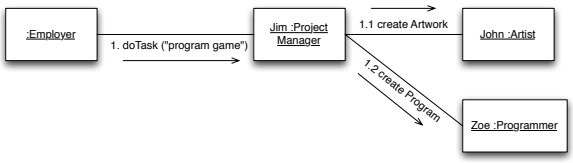


- We can then indicate this behaviour in a class diagram



Communication diagrams

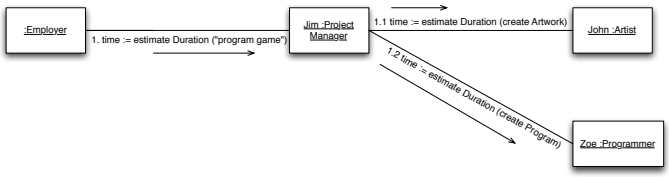
- Fullfilling a use case may result in a cascade of communication between different objects



- In this example each step is completed before the next is started.

Communication diagrams

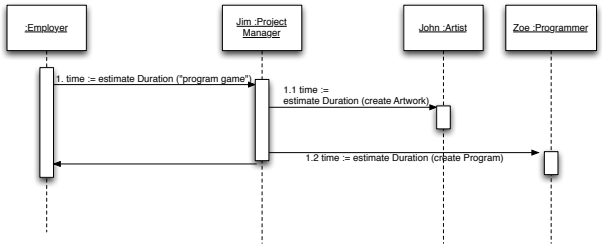
- The communication between objects can be two way...



- Jim will request estimates from John and Zoe, sum them and return the total to his employer.
- Summing these times is not shown - this is internal to Jim.
- We are only showing the communication *between* objects

Sequence diagrams

- Sequence diagrams show much the same information as a communication diagram, but they emphasise ordering...



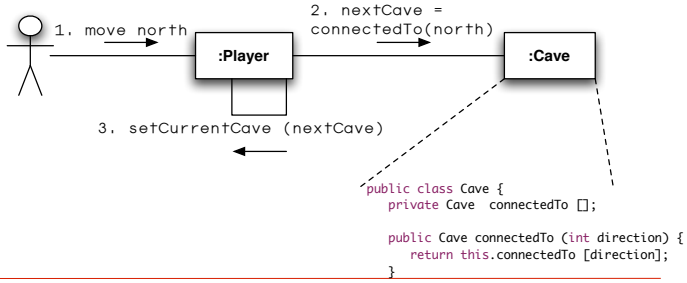
- The vertical boxes are the times that the person is busy processing the request.

Why interaction diagrams?

- We use them to explore
  - how use cases are achieved
  - what objects are involved
  - identify what behaviours we need to allocate to objects

Collaboration diagrams

- Collaboration diagrams add method calls, using numbering to show the sequence they are called.
- This diagram clearly shows the steps taken in moving caves.
- What happens if there is no cave to the north of the current one?




```
public class Cave {
    private Cave  connectedTo [];

    public Cave connectedTo (int direction) {
       return this.connectedTo [direction];
    }
```


Links in collaboration diagrams

- Like object models, links in collaboration diagrams are instances of relationships.
- Some links may be transient
  - they may only last for the duration of the collaboration, and not be shown on the class diagram.
- Each link is classified as belonging to a particular stereotype

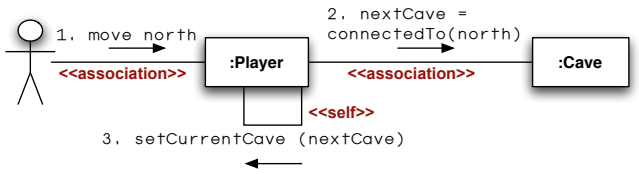
Links in collaboration diagrams

- The full list of *stereotypes* for links in interaction diagrams is..

Class Diagram	Collaboration diagram
class	object
association	*association link*
-	*parameter link*
-	*self link*
-	*local link*
-	*global link*

Links in collaboration diagrams

- Previous adventure game example more fully labeled…



Links in collaboration diagrams

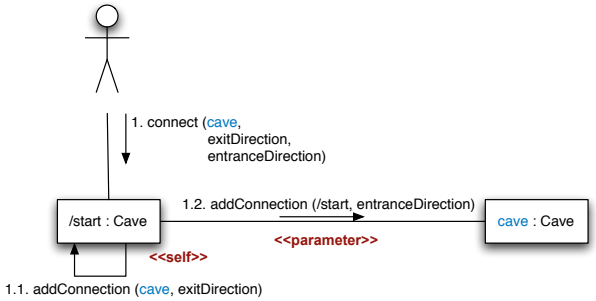
- This adventure game program sample shows objects interacting to create the initial map of the cave.
- The next slide shows the corresponding communication diagram.

```
Cave startCave = new Cave ("start cave");
 startCave.addItem (lampA);
startCave.addItem (lampB);

Cave a = new Cave ("Northern cave");
 a.addItem (gold);

Cave b = new Cave ("Eastern cave");
Cave c = new Cave ("Western Cave");
 c.addItem (oil);

// make connections
startCave.connect (a, Directions.north, Directions.south);
startCave.connect (b, Directions.east,  Directions.west);
startCave.connect (c, Directions.west,  Directions.east);
```

```
public class Cave {
   private Cave  connectedTo [];
   private String description;
   private Vector items;
   private boolean lit;

   //------------------------------------------------
   public void connect (Cave cave, int exitDirection, int entranceDirection) {
      // ensure connections go both ways
      this.addConnection (cave, exitDirection);
      cave.addConnection (this, entranceDirection);
   }
```

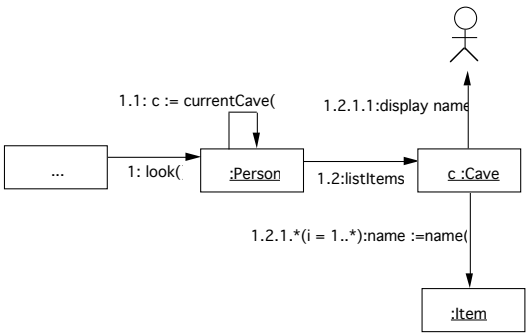
Links in collaboration diagrams

- This shows how the start cave is informed which cave to link to via a parameter.
- Normally associations in UML are bidirectional - however in Java (which is what this example is based on) is not, and we need to link in both directions.



Collaboration diagram

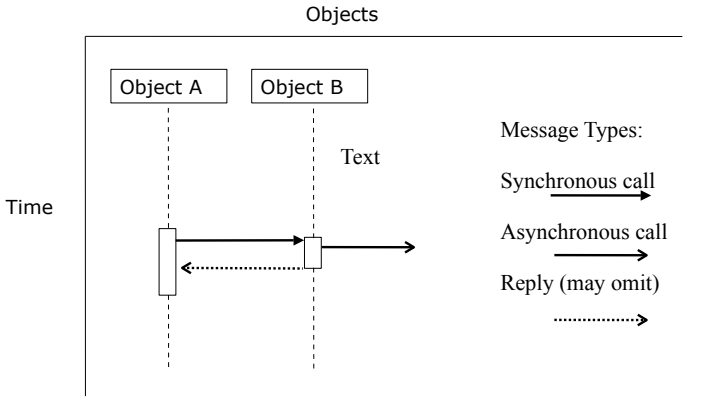
- The adventure game's look command could be drawn as follows…



Sequence Diagram

- A sequence diagram links participants to use-cases.
- A sequence diagram is an Interaction Diagram that emphasizes the time ordering of messages.
- Graphically, a sequence diagram is a table that shows participants arranged along the X axis and messages, ordered in increasing time, down the Y axis. These messages indicate the activation of an operation.
- The participant that initiates the action is placed on the left and other increasingly subordinate participants to the right.

Sequence Diagram



Sequence diagrams...

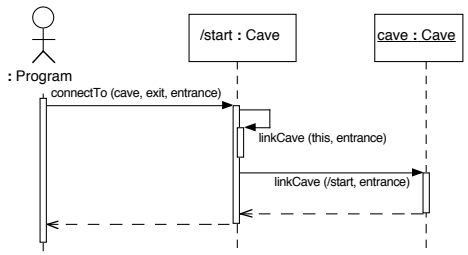
- Each message is placed along the Y axis between the participants involved in order of increasing time from top to bottom.
- Each participant has a lifeline, a vertical dashed line that represents the existence of the object over time.
- An participant may appear in many (or even all) sequence diagrams.
- From these sequence diagrams we can build up the activity of a participant (and therefore the methods needed in the class)
- Sequence diagrams only show one particular sequence of events (i.e. a response to a particular situation)

Sequence Diagrams - contd.

- Messages can be informal text messages, or formal operations (as recorded on the class diagram).
- Typically you will begin with a relatively informal diagram, work out what operations are needed, and then replace the informal messages with formal operations.
- Sequence diagrams only show the interaction between objects, not what happens inside objects.
  - There may be complexity hidden inside the objects that is not shown on a diagram.
  - Similar to listening to conversations in a room - you hear people talking to each other. You don't hear what they are thinking.

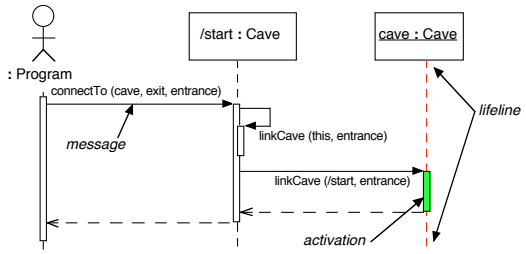
Sequence diagrams

- The collaboration diagram shown previously translated to a sequence diagram.



Sequence diagrams

- The parts of the sequence diagram

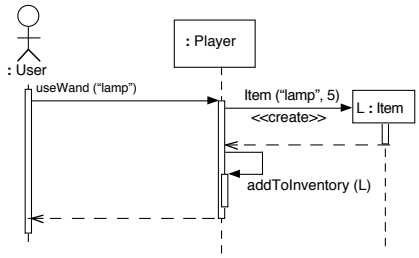


Sequence diagrams - object creation

- We can show items being created.
- Their lifeline starts midway through the sequence diagram, rather than being in existence at the start of the sequence.
- Consider a magic wand that can create items in the adventure game

Sequence diagrams - object creation

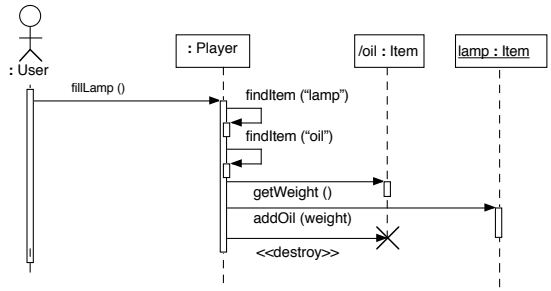
- You can specify a constructor, or use the action "create"



What does the activity underneath ***L: Item*** represent?

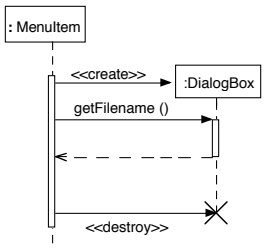
Sequence diagrams - object deletion

- For object deletion you use the action "destroy"



Sequence diagrams - object creation and deletion

- Another example...



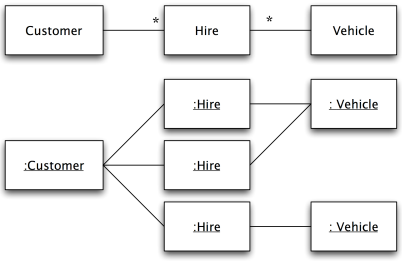
Do we need <<destroy>> in Java?

Sequence diagrams - Iteration

- Sometimes we have to send the same message to a number of items of the same type.
- We add multiplicity symbols to the classifier role (the box at the top of a sequence diagram life line).
- Methods are also annotated to indicate they are being called multiple times.

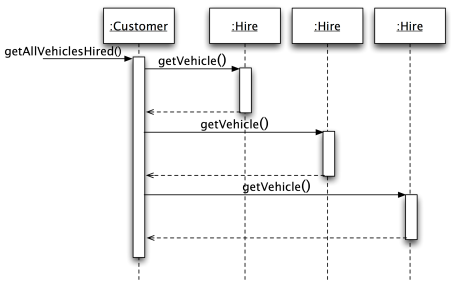
Sequence diagrams - Iteration

- If we wanted to list all the cars that a customer had borrowed we could examine a particular scenario, and simply show sending a message to each item.



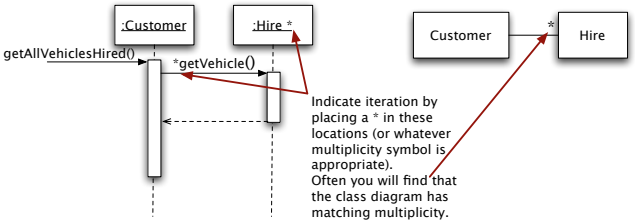
Sequence diagrams - Iteration

- Showing the same message to each item.
- As each Hire object has a link to a Vehicle, it can return that link immediately.



Sequence diagrams - Iteration

- You can show a more general scenario using iteration.
- This moves the diagram away from a modeling just a single scenario however - it now models a groups of scenarios.

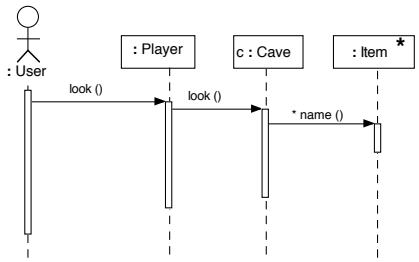


Sequence diagrams - Iteration

- Adventure game - consider the "look" command.
- An algorithm can be derived for it
  - 1.1 Find the cave the player is in
  - 1.2 For each item in the cave
    - 1.2.1 Get the items name
    - 1.2.2 Display the name
- This number scheme corresponds to the numbering in a collaboration diagram.

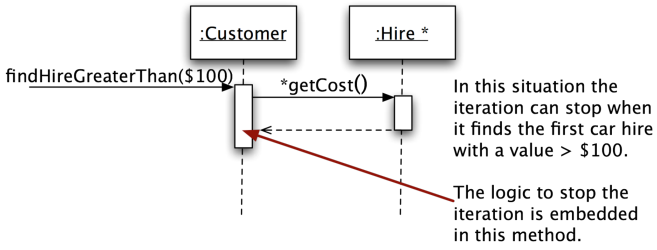
Sequence diagrams - Iteration

- Diagram for a "look" command



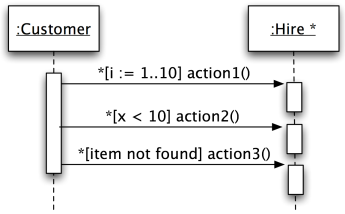
Sequence diagrams - Iteration

- Iteration does not need to proceed through all of the items in an association. The iteration could stop when a searched-for item is found.



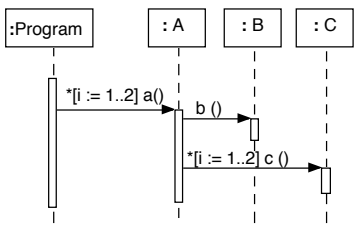
Sequence diagrams - Iteration

- You can specify at iteration schema that explicitly states when the iteration ends.
- You can compose any definite or indefinite iteration.
- Indefinite iteration will run while the condition is true
- The syntax inside the [ ] is not specified - you can use psuedocode, the implementation language, or anything else.



Sequence diagrams - Nested Iteration

- Nested iteration can be shown on a diagram



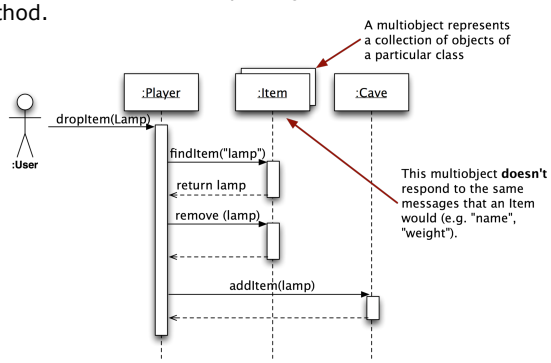
This will result in the sequence of calls of abccabcc

Sequence diagrams - Multiobjects

- To reduce the amount of repetition in sequence diagrams we introduced iteration.
- The next step is to deal with groups, or sets of objects.
- Groups have specific behaviours
  - Insert, Find, Delete, etc
  - set based operations (union, intersection etc)
- When there is a multiplicity of more than 1 in an association, we sometimes want to deal with the objects as a group.
- A multiobject is the UML name for a group of objects.

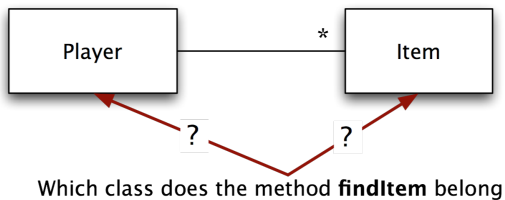
Sequence diagrams - Multiobjects

- Note that the message "findItem" is not a repeated message.
- Any iteration across multiple objects exists in the "findItem" method.



Sequence diagrams - Multiobjects

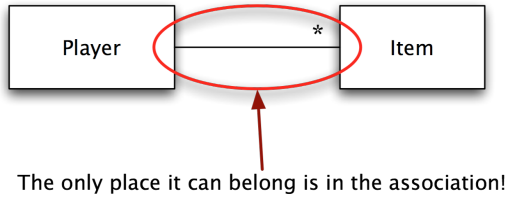
- So where should the methods that are called on a multiobject go?



Which class does the method **findItem** belong in?

Sequence diagrams - Multiobjects

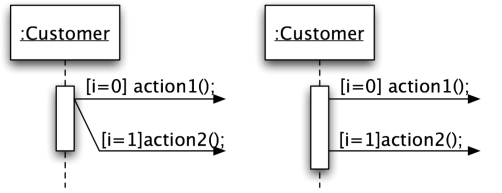
- A multiobject is actually an instance of the association between two classes.
- The multiobject and its methods end up being the implementation of an association in the design stage.



The only place it can belong is in the association!

Sequence diagrams - selection

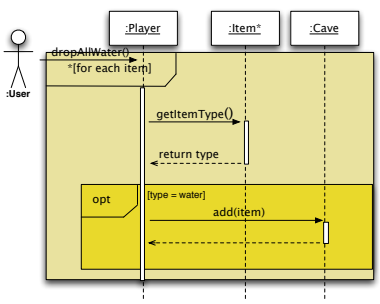
- Sequence diagrams can specify alternative paths, depending upon a condition.
- It can make diagrams very hard to read - you are probably better using activity diagrams instead.



if/else statement Two consecutive if statements

Sequence diagrams - Boxing commands

- Sometimes you want to include a number of statements inside an iteration or selection
- This is shown by "boxing up" the method calls controlled by the iteration or selection.



Other Interaction Frames

- opt
  - Optional – the same as an alt with only one trace
- par
  - Each trace is run in parallel
- region
  - This is a critical region – only one thread may be executing within the fragment
- neg
  - The trace shows an invalid interaction
- ref
  - The trace makes reference to another interaction model. Input and output parameters may be defined
- sd
  - You can surround the entire sequence diagram, but this adds little value

Problems with sequence diagrams

- Conditional/Repetitious behaviour hard to model
- Hard to display interactions with multiple objects
- Many required for each use case
- Some behaviours are better specified using a relational algebra (e.g. SQL)

Notes

- When to use interaction diagrams
  - When you need to show behaviour of multiple objects in the same Use Case
  - Sequence vs. Communication depends on what you get the most out of for that use case
    - each designer will probably lean towards one or the other as a 'favourite' for most situations
  - For behaviour over a number of use cases, state or activity diagrams are probably better